

Assignment-2

Name of the Student	S.Abarnasri
Register Number	820519205001
Maximum Mark	2 Mark

Questions:

Create User table with user email, username, roll number, password.

1. Perform UPDATE, DELETE Queries with user table.
2. Connect python code to db2.
3. Create a flask app with registration page, login page and welcome page. By default load the registration page once the user enters all the fields store the data in database and navigate to login page authenticate user username and password. If the user is valid show the welcome page.

Solutions:

1. Creating user table with user email, username, roll number, password.

The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: <https://cloud.ibm.com/console/region/US-south/instances/53a462f1-0829-015a66ba1126/tables>. The page title is "IBM Db2 on Cloud".

The navigation bar includes links for Load Data, Load History, Tables, Views, Indexes, Aliases, MQTs, Sequences, and Application objects. The "Tables" tab is selected.

A search bar at the top left of the main content area is labeled "Find schemas or tables". Below it, a table lists the available tables:

Name	Schema	Properties
STUDENT	MCL23449	—

At the bottom of this table, it says "Total: 1, selected: 1".

To the right of the table list is a "Table definition" panel for the "STUDENT" table. It shows the table's structure with the following columns:

Name	Data type	Nullable	Length	Scale
USERNAME	CHAR	N	8	0
EMAIL	VARCHAR	Y	32	0
Roll number	VARCHAR	Y	32	0
PASSWORD	VARCHAR	Y	32	0

Below the table definition, there is a "View data" button.

The Windows taskbar at the bottom shows the system clock as 10:41 on 19-10-2022.

Device Details: IBM Cloud

IBM Db2 on Cloud

https://cloud.ibm.com/devops/visualizations/db2-console/visualizations/33f4e3a4c52f100210-55a49a0349a2...

IBM Db2 on Cloud

Data objects

My script

Filter objects

NO123456

Tables

STUDENT

Views

MQTs

Aliases

Namespaces

Untitled - 1

Serial assistant

Run all

```
1 insert into student values('ayush',3,8,1);
2 insert into student values('anu',9,4,6);
3
```

History

Find history

Script	Date	Status	Runtime	
Untitled - 1	Oct 19, 2022 6:58:12 AM	Success	0.013 s	1
insert into student values('ayush',3,8,1)		Success	0.007 s	1
insert into student values('anu',9,4,6)		Success	0.000 s	1
Untitled - 1	Oct 19, 2022 6:57:27 AM	Failure	0.002 s	1
Untitled - 1	Oct 19, 2022 6:55:48 AM	Failure	0.020 s	1

Sign here to search

16.10.2022

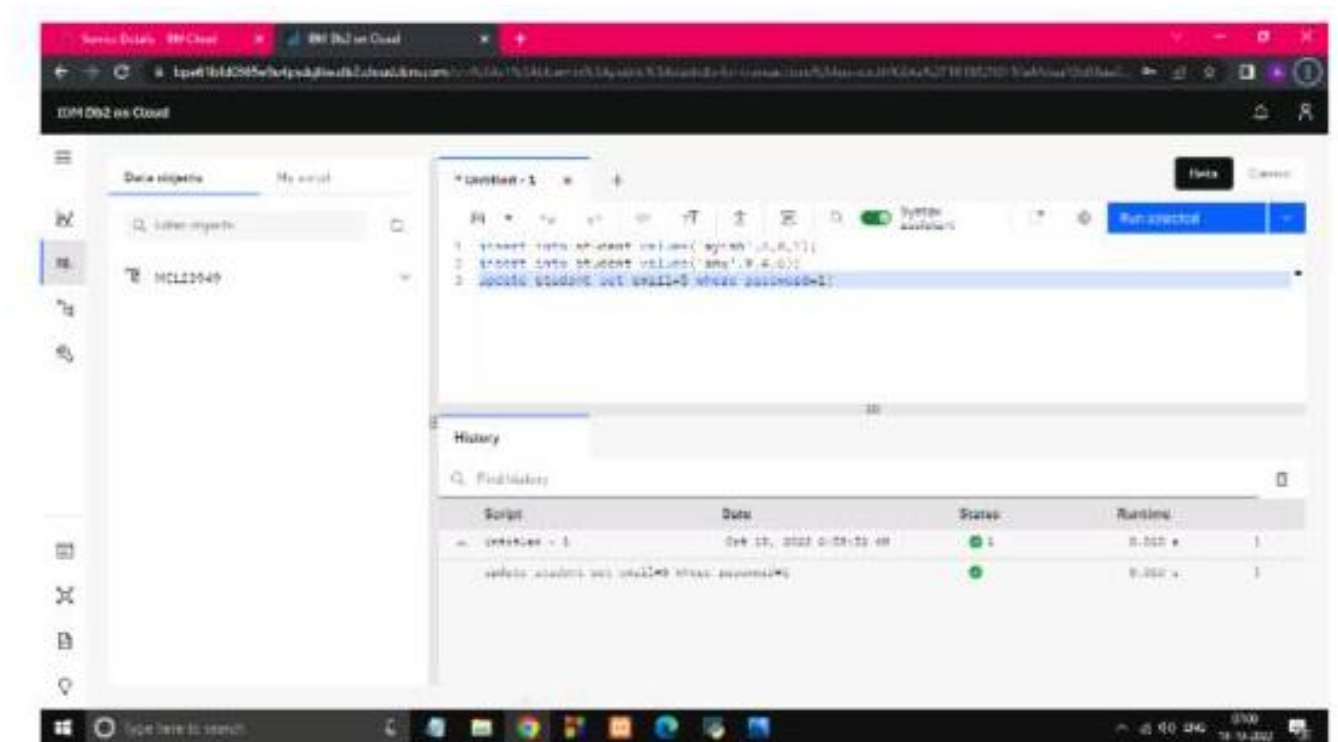
Output:

The screenshot shows a web application interface with a dark theme. At the top, there's a navigation bar with tabs for 'Schema Details', 'DB Chat', and 'DB DDL on Cloud'. Below this is a breadcrumb trail: 'Export to CSV' > 'MCL23949 STUDENT'. The main content area displays a table with four columns: 'USERNAME', 'EMAIL', 'Roll number', and 'PASSWORD'. The table contains three rows of data. To the right of the table, there are two buttons: 'Back' and 'Export to CSV'. The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 10:40 on 19-10-2022.

USERNAME	EMAIL	Roll number	PASSWORD
anu	S	4	6
ayul	S	8	5
ayul	S	8	5

2. Performing UPDATE, DELETE Queries with user table

UPDATE:



OUTPUT:

IBM Db2 on Cloud

Load Data Load History Tables Views Indexes Aliases MQ's Sequences Application objects

MCL23949.STUDENT

Export to CSV

USERNAME	EMAIL	Roll number	PASSWORD
amr	0	4	4
ayush	1	8	0
ayush	1	8	1

DELETE:

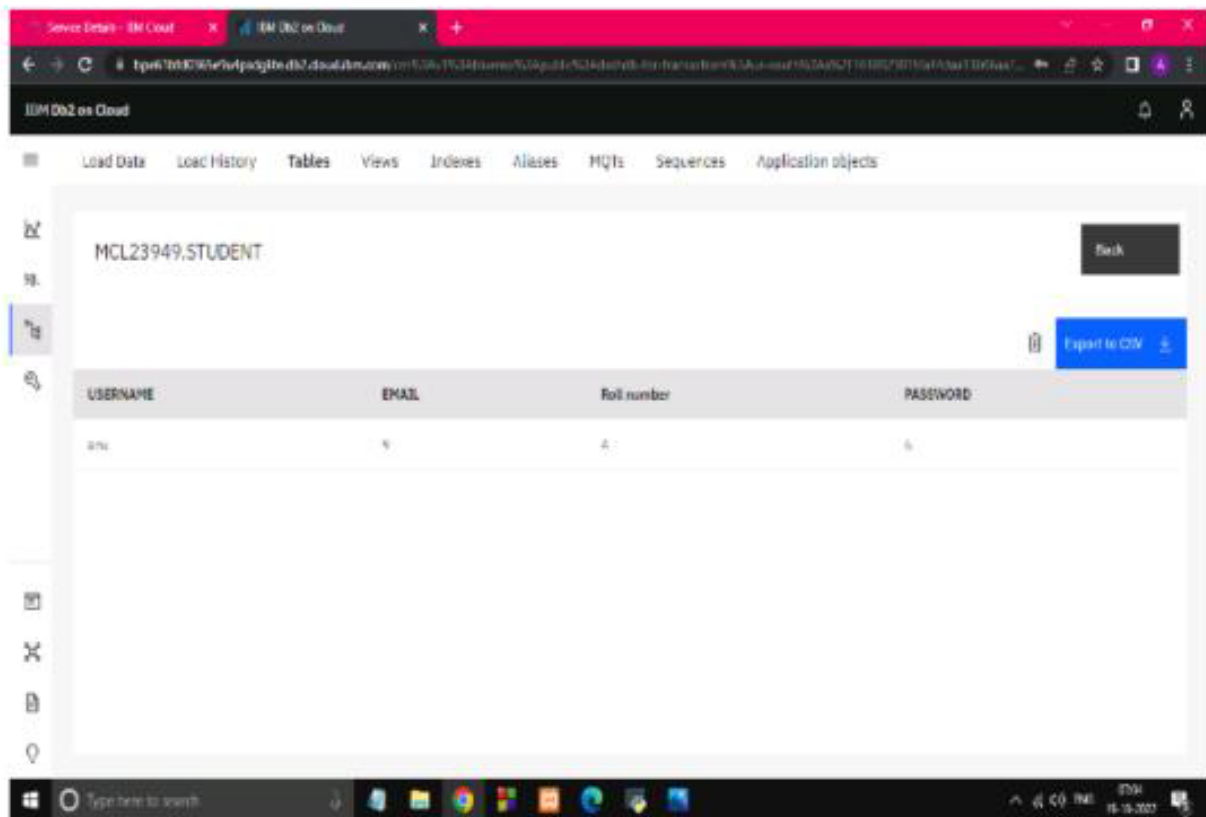
The screenshot displays the IBM DB2 on Cloud web interface. On the left, a sidebar shows the 'Data objects' tab with a search bar and a list containing 'MCL21040'. The main area is divided into two panes. The top pane, titled 'Untitled - 1', contains a SQL script with four lines:

```
1 insert into student values('99999',9.9,1);
2 insert into student values('999',9.9,1);
3 update student set email='S' where password='1';
4 delete from student where email='S';
```

 The bottom pane, titled 'History', shows a table of executed scripts. The table has columns for 'Script', 'Date', 'Status', and 'Runtime'. It lists three previous executions of the script, all with a status of 'Success' and a runtime of 0.015 seconds.

Script	Date	Status	Runtime
insert into student values('99999',9.9,1); insert into student values('999',9.9,1); update student set email='S' where password='1'; delete from student where email='S';	Oct 18, 2023 7:04:09 AM	Success	0.015 s
insert into student values('99999',9.9,1); insert into student values('999',9.9,1); update student set email='S' where password='1'; delete from student where email='S';	Oct 18, 2023 6:59:15 AM	Success	0.015 s
insert into student values('99999',9.9,1); insert into student values('999',9.9,1); update student set email='S' where password='1'; delete from student where email='S';	Oct 18, 2023 6:58:15 AM	Success	0.015 s

OUTPUT:



3.Connect python code to db2.

```
import ibm_db

conn=

ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb
7e483086.clogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURI
TY=SSL;SSLServerCertificate=DigiCertGlobalRoot.crt;UID=mcl23949;PWD=dmUiv
2o6zjDMw0Ea",)

print(conn)

print("connection successful...")
```

4.Creating a flask app with registration page, login page and welcome page.

```
# Store this code in 'app.py' file
```

```

from flask import Flask, render_template, request, redirect, url_for, session

import ibm_db

import re

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf
9-0fbb7e483086.clogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;
SECURITY=SSI;SSLServerCertificate=DigiCertGlobalRoot.crt;UID=mcl23949;PWD=
dmUiv2o6zjDMw0Ea",)

app = Flask(__name__)
app.secret_key='a'

conn = ibm_db.connect()

@app.route('/')
@app.route('/login', methods =['GET', 'POST'])
def login():
    msg = "

    if request.method == 'POST' and 'username' in request.form and 'password' in
    request.form:

        username = request.form['username']
        password = request.form['password']

        sql=('SELECT * FROM users WHERE username = % s AND password = % s',
        (username, password, ))

        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fet_assoc(stmt)

```

```
print(account)

if account:
    session['loggedin'] = True
    session['id'] = account['id']
    session['username'] = account['username']
    msg = 'Logged in successfully !'
    return render_template('index.html', msg = msg)
else:
    msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
```

```
@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))
```

```
@app.route('/register', methods = ['GET', 'POST'])
def register():
    msg = ""

    if request.method == 'POST' and 'username' in request.form and 'password' in request.form and 'email' in request.form :
        username = request.form['username']
        password = request.form['password']
```

```

email = request.form['email']

sql=('SELECT * FROM users WHERE username = % s AND password = % s',
(username, password, ))

stmt = ibm_db.prepare(conn,sql)

ibm_db.bind_param(stmt,1,username)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

if account:

    msg = 'Account already exists !'

elif not re.match(r'^@]+@[^@]+\.[^@]+', email):

    msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

    msg = 'Username must contain only characters and numbers !'

elif not username or not password or not email:

    msg = 'Please fill out the form !'

else:

    insert_sql=('INSERT INTO accounts VALUES (NULL, % s, % s, % s)', (username,
password, email, ))

    prep_stmt = ibm_db.prepare(conn,insert_sql)

    ibm_db.bind_param(prepare_stmt,1,username)

            ibm_db.bind_param(stmt,2,password)

            ibm_db.bind_param(stmt,3,email)

            ibm_db.execute(prepare_stmt)

    msg = 'You have successfully registered !'

    elif request.method == 'POST':

```

```
msg = 'Please fill out the form !'
return render_template('register.html', msg = msg)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

register.html:

```
<html>
<head>
<meta charset="UTF-8">
<title> Register </title>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body style="background-color:powderblue;"><br><br><br><br><br>
<div align="center">
<div align="center" class="border">
<div class="header">
<h1 class="word">Register</h1>
</div><br><br><br>
<h2 class="word">
<form action="{{ url_for('register') }}" method="post">
<input id="username" name="username" type="text" placeholder="Enter Your
Username" class="textbox"/><br><br>
<input id="password" name="password" type="password" placeholder="Enter
Your Password" class="textbox"/><br><br>
<input id="email" name="email" type="text" placeholder="Enter Your Email ID"
class="textbox"/><br><br>
```

```

<input type="submit" class="btn" value="Sign Up"></br>
</form>

</h2>

<p class="bottom">Already have an account? <a class="bottom"
href="{{url_for('login')}}"> Sign In here</a></p>

</div>

</div>

</body>

</html>

```

login.html:

```

<html>

<head>

<meta charset="UTF-8">

<title> Login </title>

<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body></br></br></br></br></br>

<div align="center">

<div align="center" class="border">

<div class="header">

<h1 class="word">Login</h1>

</div></br></br></br>

<h2 class="word">

<form action="{{ url_for('login') }}" method="post">

```

```

<input id="username" name="username" type="text" placeholder="Enter Your
Username" class="textbox"/></br></br>

<input id="password" name="password" type="password" placeholder="Enter
Your Password" class="textbox"/></br></br></br>

<input type="submit" class="btn" value="Sign In"></br></br>

</form>

</h2>

<p class="bottom">Don't have an account? <a class="bottom"
href="{{url_for('register')}}"> Sign Up here</a></p>

</div>

</div>

</body>

</html>

```

Index.html:

```

<html>

<head>

<meta charset="UTF-8">

<title> Index </title>

<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body></br></br></br></br></br>

<div align="center">

<div align="center" class="border">

<div class="header">

<h1 class="word">Index</h1>

```

```
</div></br></br></br>
```

```
<h1 class="bottom">
```

```
Hi {{session.username}}!!</br></br> Welcome to the index page...
```

```
</h1></br></br></br>
```

```
<a href="{{ url_for('logout') }}" class="btn">Logout</a>
```

```
</div>
```

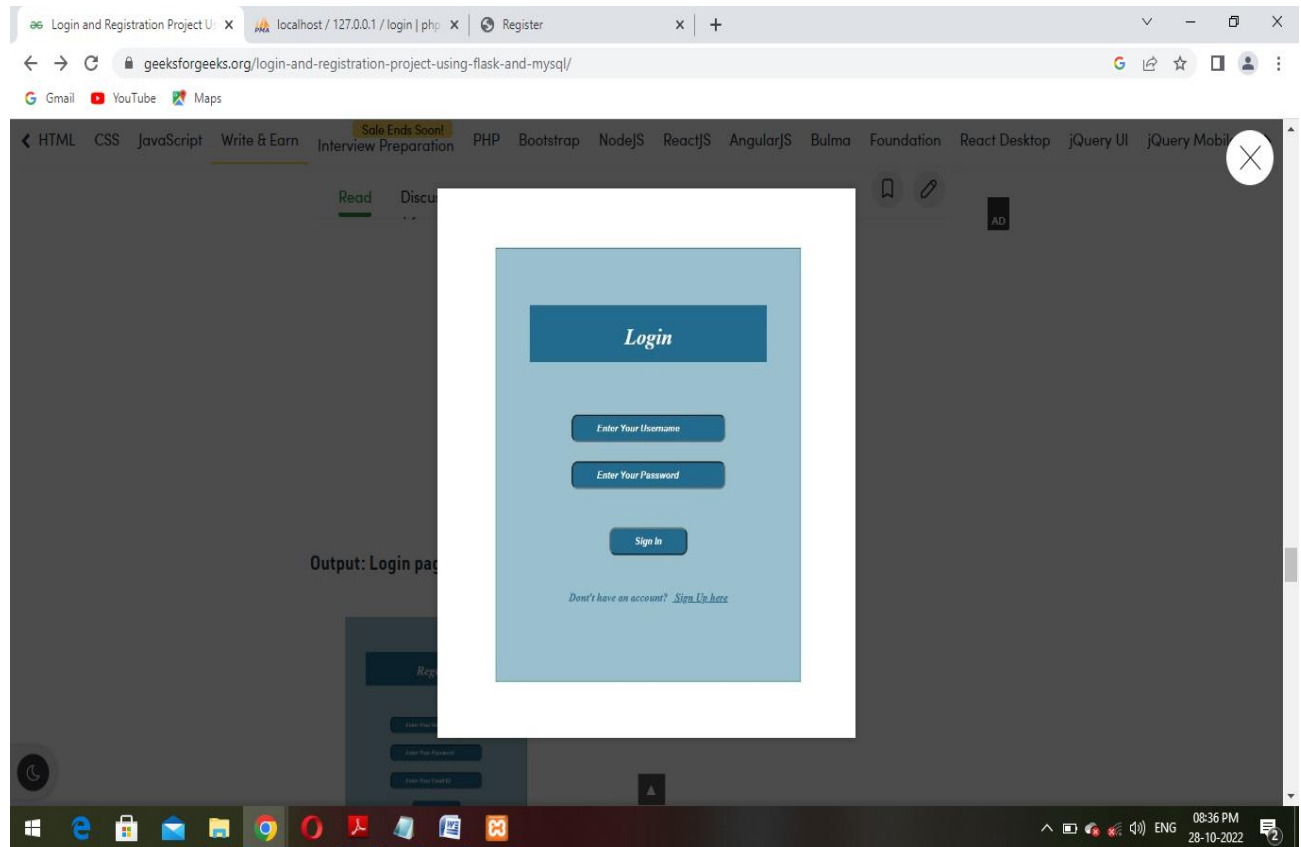
```
</div>
```

```
</body>
```

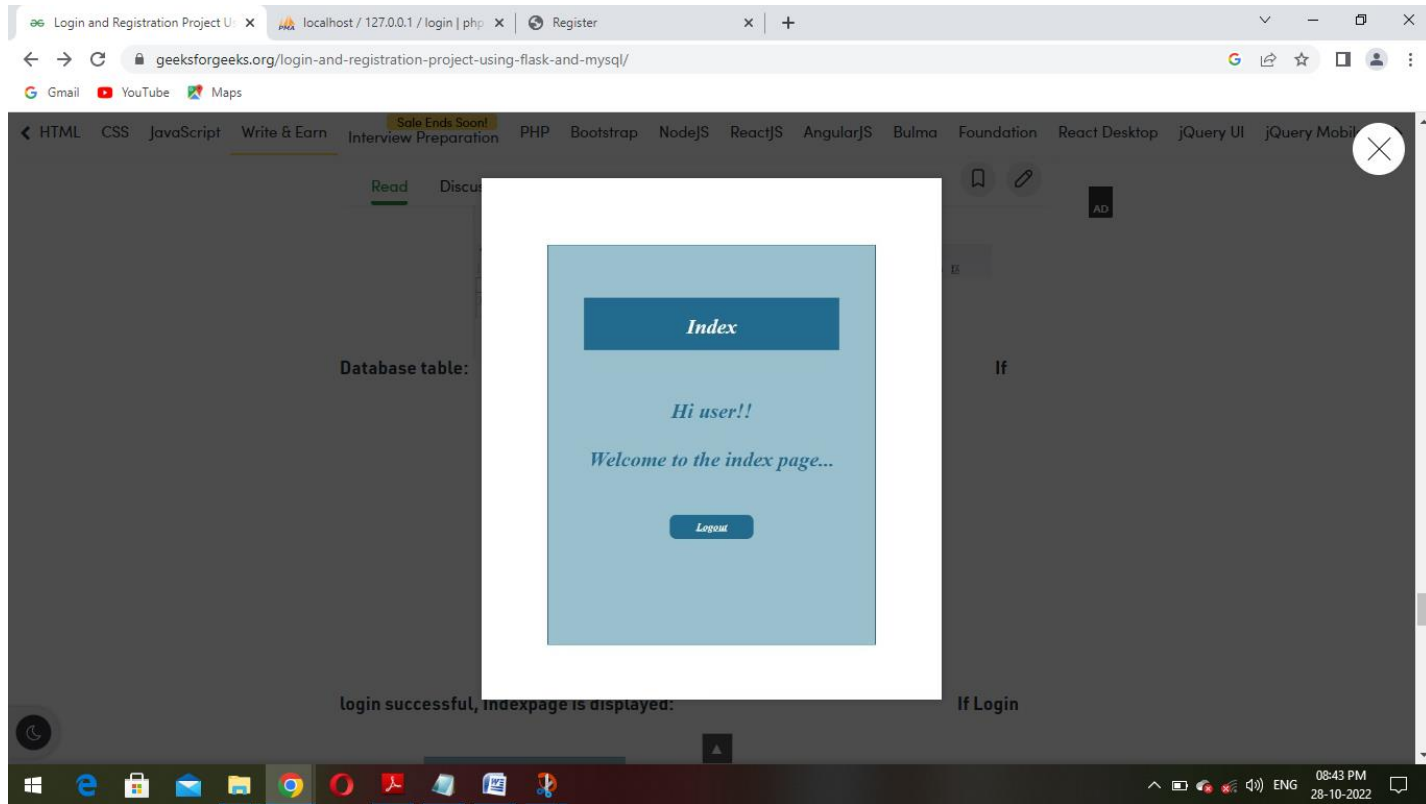
```
</html>
```


OUTPUT

Register page:



Welcome Page:



Login Page:

